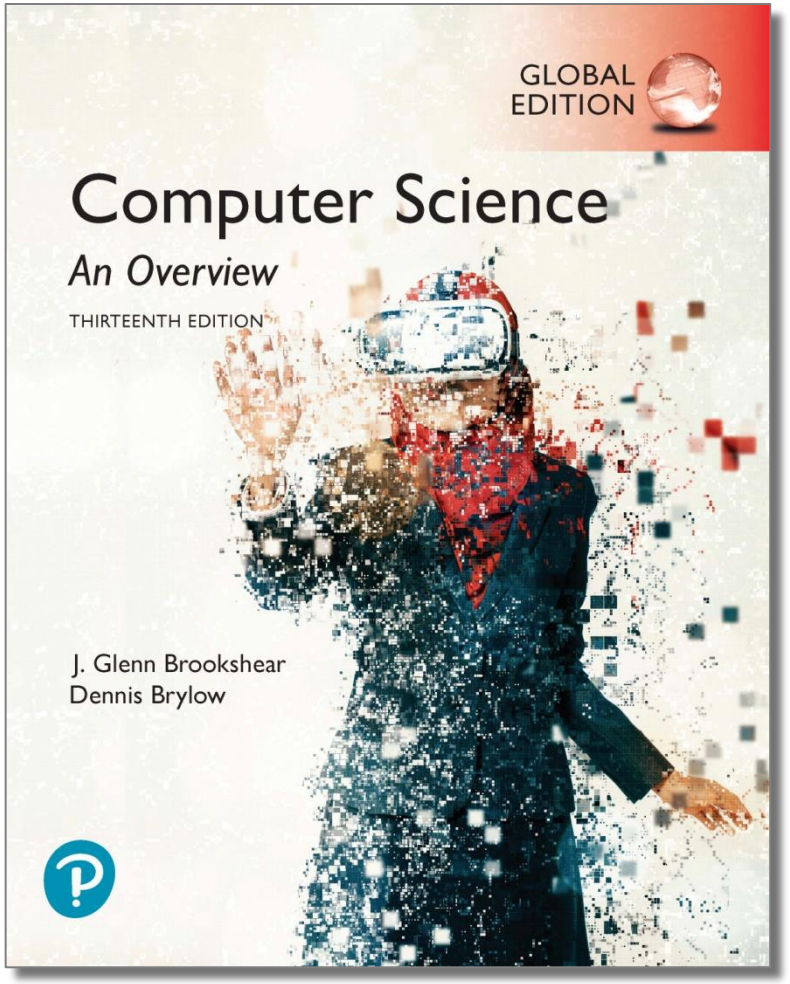
Computer Science An Overview

13th Edition, Global Edition



Chapter 1

Data Storage

Chapter 1: Data Storage

- 1.1 Bits and Their Storage
- 1.2 Main Memory
- 1.3 Mass Storage
- 1.4 Representing Information as Bit Patterns
- 1.5 The Binary System

Chapter 1: Data Storage (continued)

- 1.6 Storing Integers
- 1.7 Storing Fractions
- 1.8 Data and Programming
- 1.9 Data Compression
- 1.10 Communications Errors

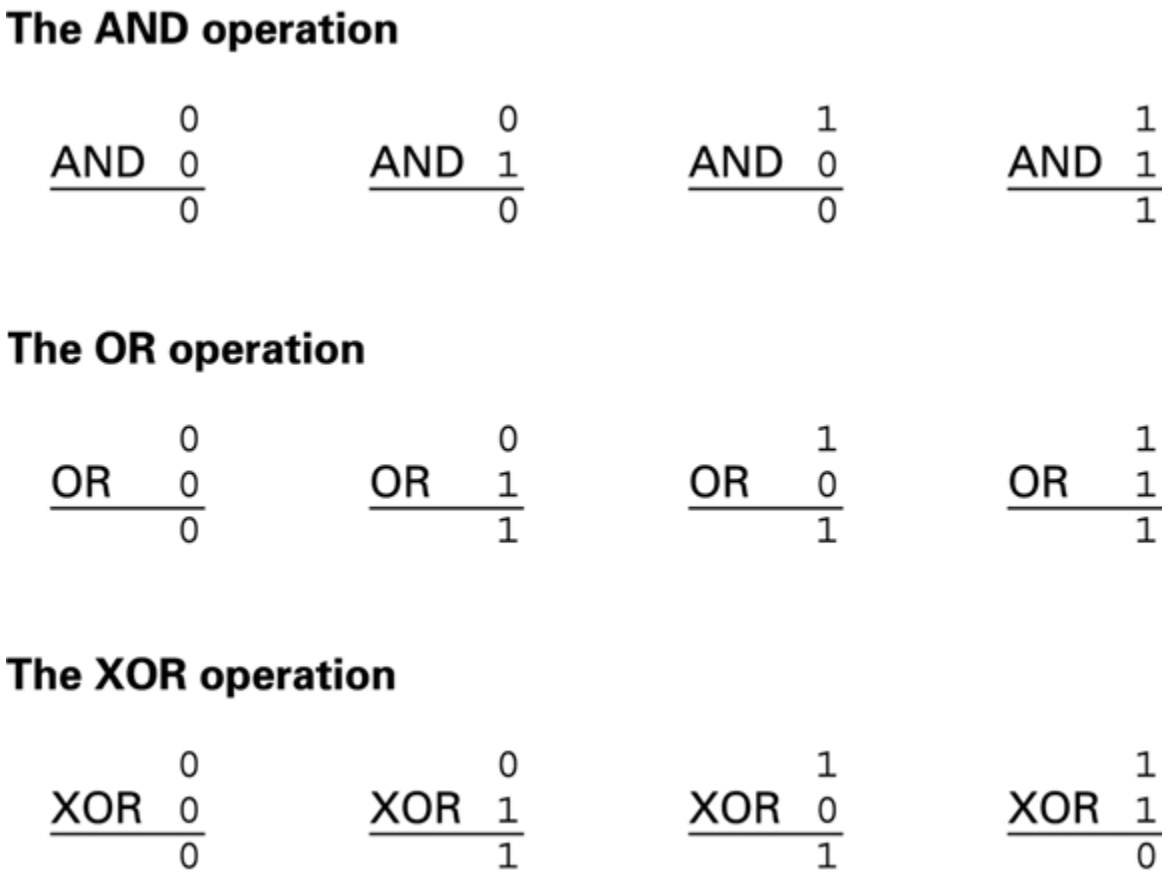
1.1 Bits and Their Storage

- **Bit:** Binary Digit (0 or 1)
- Bit Patterns are used to represent information
  - Numbers
  - Text characters
  - Images
  - Sound
  - And others

Boolean Operations

- **Boolean Operation:** An operation that manipulates one or more true/false values
- Specific operations
  - AND
  - OR
  - XOR (exclusive or)
  - NOT

Figure 1.1 The possible input and output values of Boolean operations AND, OR, and XOR (exclusive or)

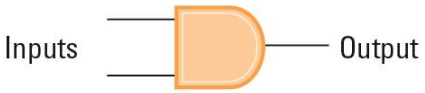


Gates

- **Gate:** A device that computes a Boolean operation
  - Often implemented as small electronic circuits called transistors
  - Can be constructed from a variety of other technologies
  - Provide the building blocks from which computers are constructed

Figure 1.2 A pictorial representation of AND, OR, XOR, and NOT gates as well as their input and output values

AND



Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

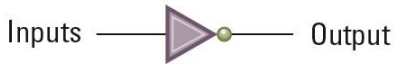
OR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR

Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

NOT



Inputs	Output
0	1
1	0

Flip-flops

- Circuits built from gates that act as a fundamental unit of computer memory
  - One input line is used to set its stored value to 1
  - One input line is used to set its stored value to 0
  - While both input lines are 0, the most recently stored value is preserved

Figure 1.3 A simple flip-flop circuit

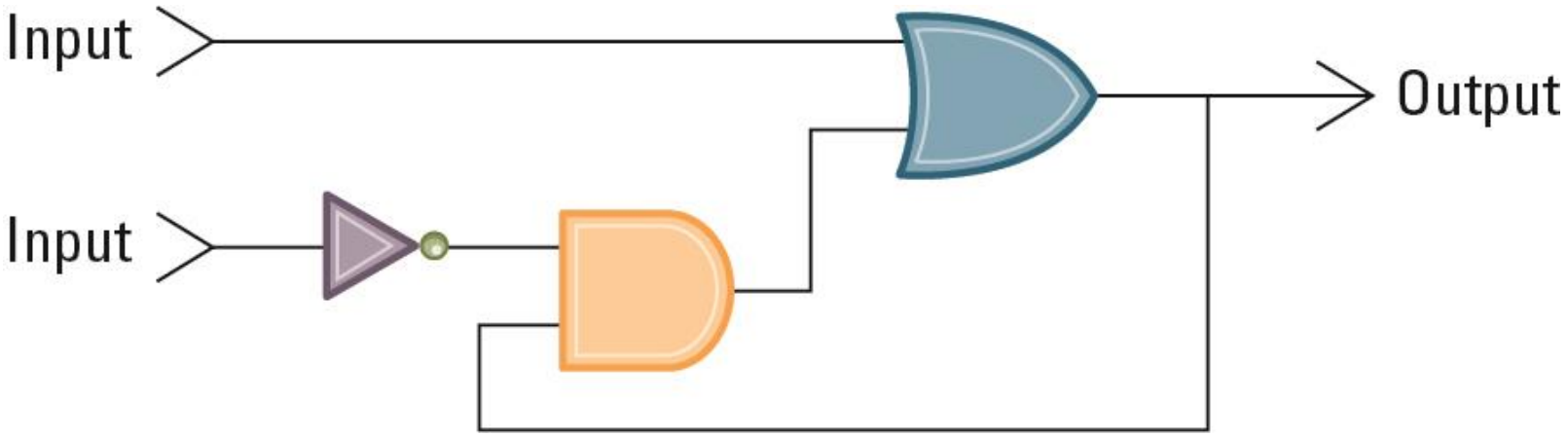


Figure 1.4 Setting the output of a flip-flop to 1

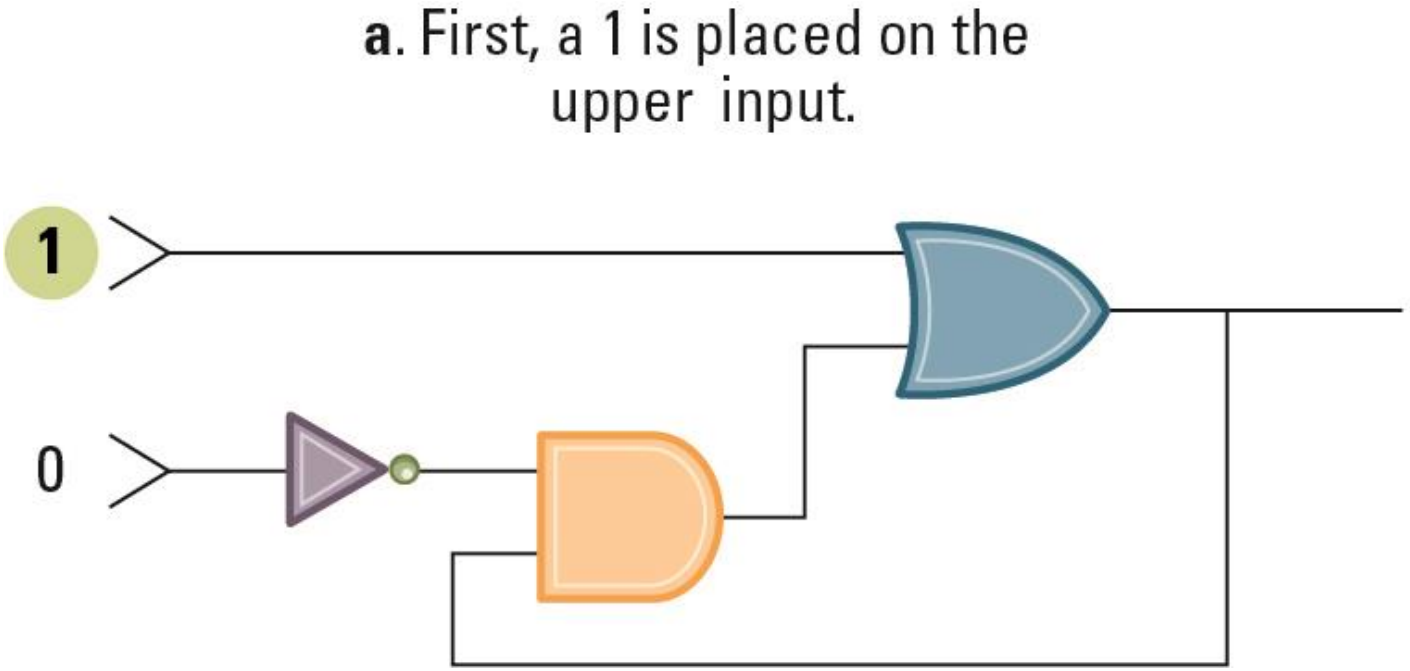


Figure 1.4 Setting the output of a flip-flop to 1 (continued)

b. This causes the output of the OR gate to be 1 and, in turn, the output of the AND gate to be 1.

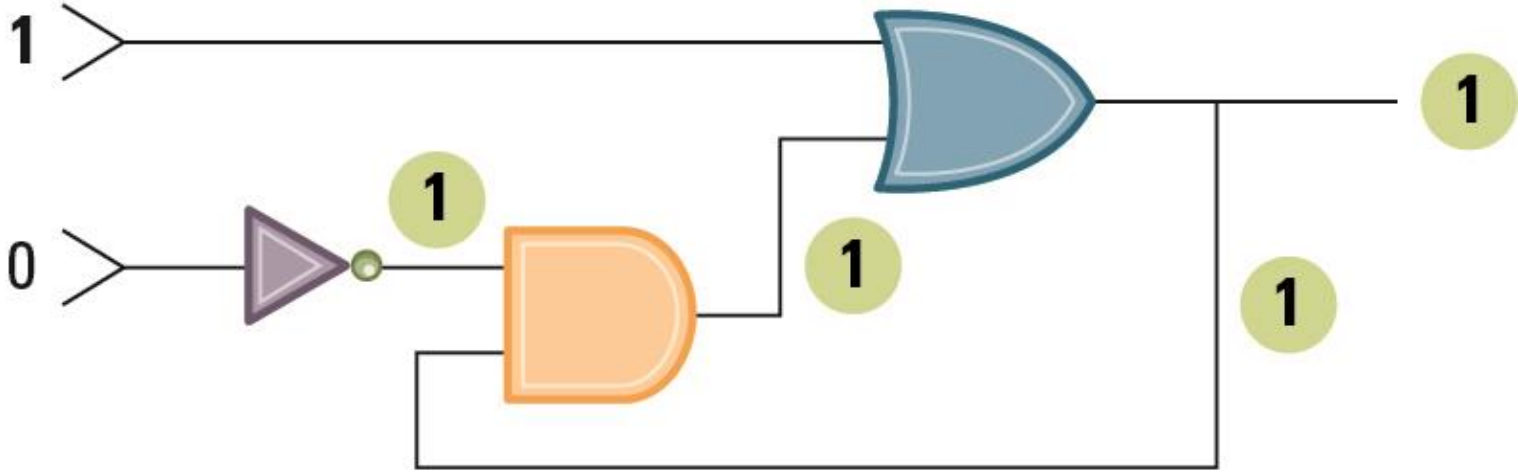


Figure 1.4 Setting the output of a flip-flop to 1 (continued)

c. Finally, the 1 from the AND gate keeps the OR gate from changing after the upper input returns to 0.

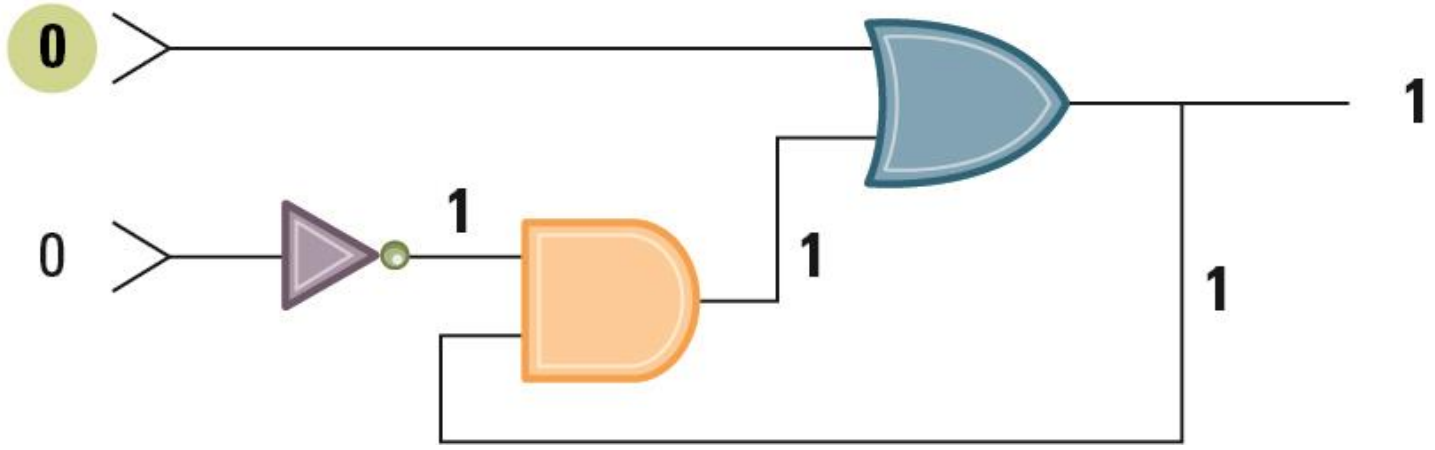


Figure 1.5 Another way of constructing a flip-flop

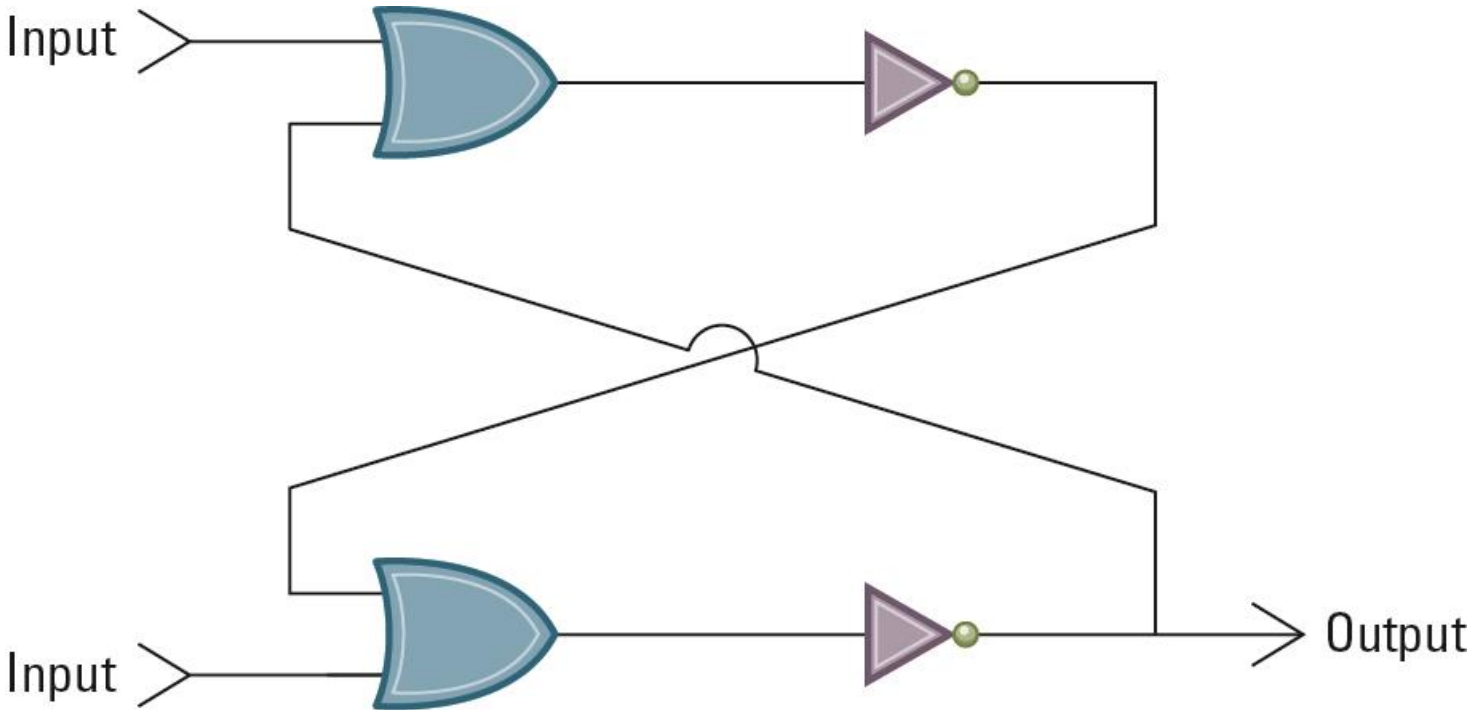
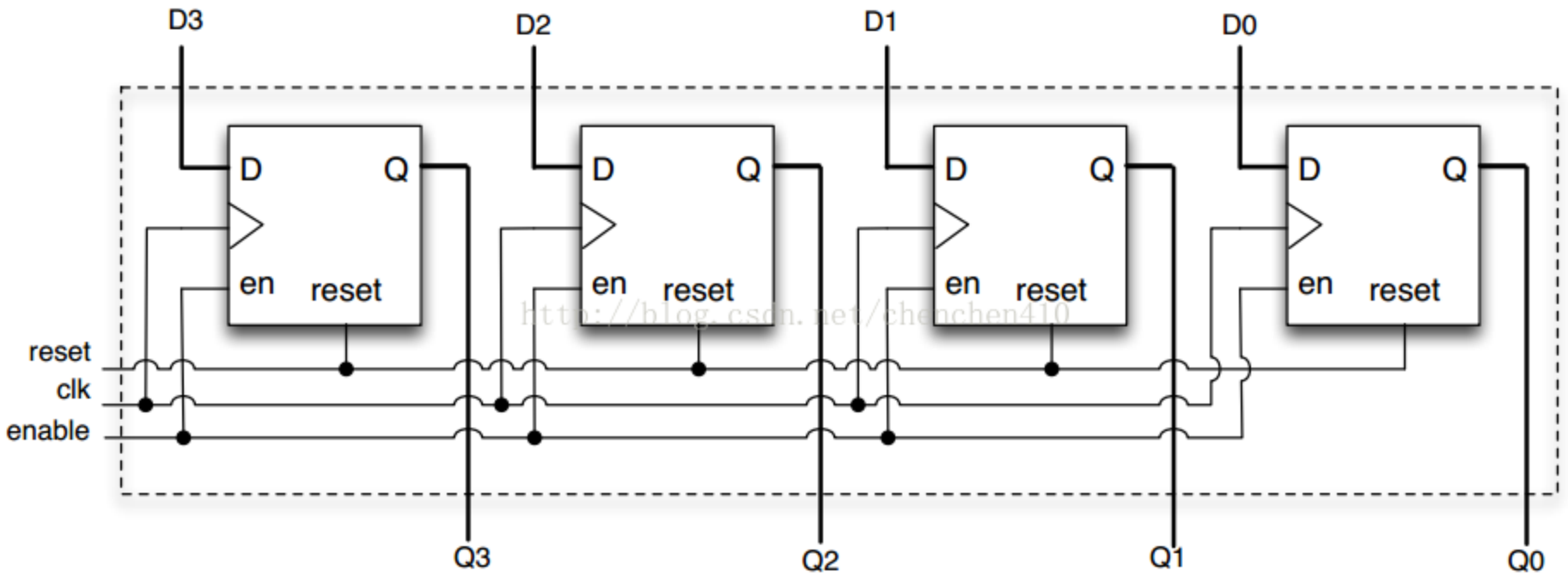


Figure 1.5 Another way of constructing a flip-flop



https://www.twblogs.net/a/5b8ad83b2b71775d1ce97a67

Hexadecimal Notation

- **Hexadecimal notation:** A shorthand notation for long bit patterns
  - Divides a pattern into groups of four bits each
  - Represents each group by a single symbol
- Example: 10110101 becomes 0xB5

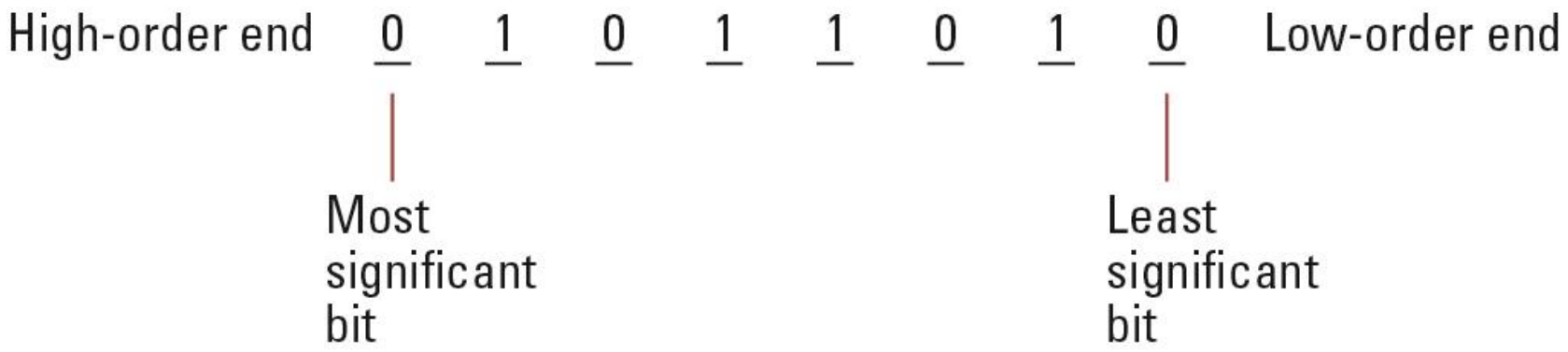
Figure 1.6 The hexadecimal coding system

Bit pattern	Hexadecimal representation
0000	0x0
0001	0x1
0010	0x2
0011	0x3
0100	0x4
0101	0x5
0110	0x6
0111	0x7
1000	0x8
1001	0x9
1010	0xA
1011	0xB
1100	0xC
1101	0xD
1110	0xE
1111	0xF

1.2 Main Memory

- **Cell:** A unit of main memory (typically 8 bits which is one **byte**)
  - **Most significant bit:** the bit at the left (high-order) end
  - **Least significant bit:** the bit at the right (low-order) end

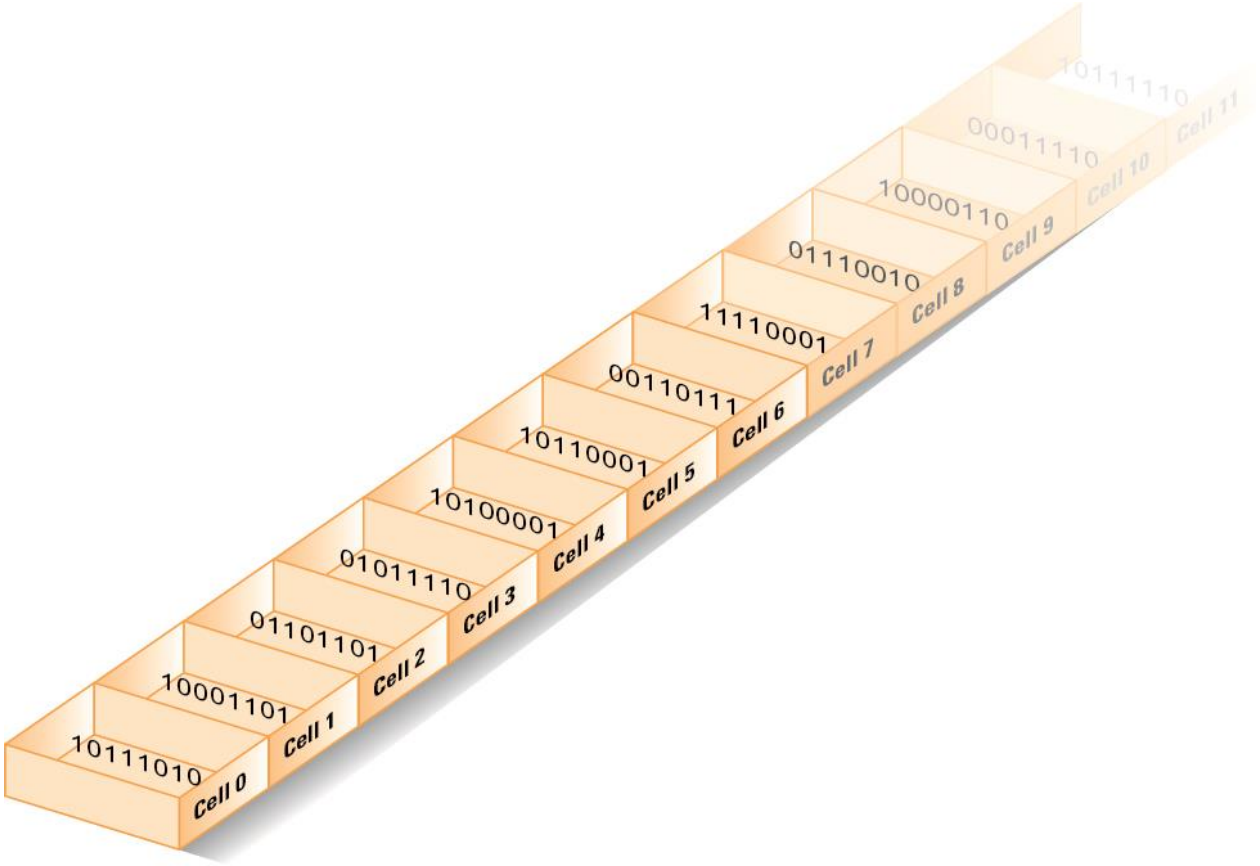
Figure 1.7 The organization of a byte-size memory cell



Main Memory Addresses

- **Address:** A “name” that uniquely identifies one cell in the computer’s main memory
  - The names are actually numbers.
  - These numbers are assigned consecutively starting at zero.
  - Numbering the cells in this manner associates an order with the memory cells.

Figure 1.8 Memory cells arranged by address



Memory Terminology

- **Random Access Memory (RAM):** Memory in which individual cells can be easily accessed in any order
- **Dynamic Memory (DRAM):** RAM composed of volatile memory

Measuring Memory Capacity

- **Kilobyte:** 2^{10} bytes = 1024 bytes
  - Example: 3 KB = 3 times1024 bytes
- **Megabyte:** 2^{20} bytes = 1,048,576 bytes
  - Example: 3 MB = 3 times 1,048,576 bytes
- **Gigabyte:** 2^{30} bytes = 1,073,741,824 bytes
  - Example: 3 GB = 3 times 1,073,741,824 bytes

1.3 Mass Storage

- Additional devices:
  - Magnetic disks
  - CDs
  - DVDs
  - Magnetic tapes
  - Flash drives
  - Solid-state drives
- Advantages over main memory
  - Less volatility
  - Larger storage capacities
  - Low cost
  - In many cases can be removed

Mass Storage Performance

- **Bandwidth**: The total amount of bits that can be transferred in a unit of time
- **Latency**: The total time between the request for data transfer and its arrival

Figure 1.9 A magnetic disk storage system

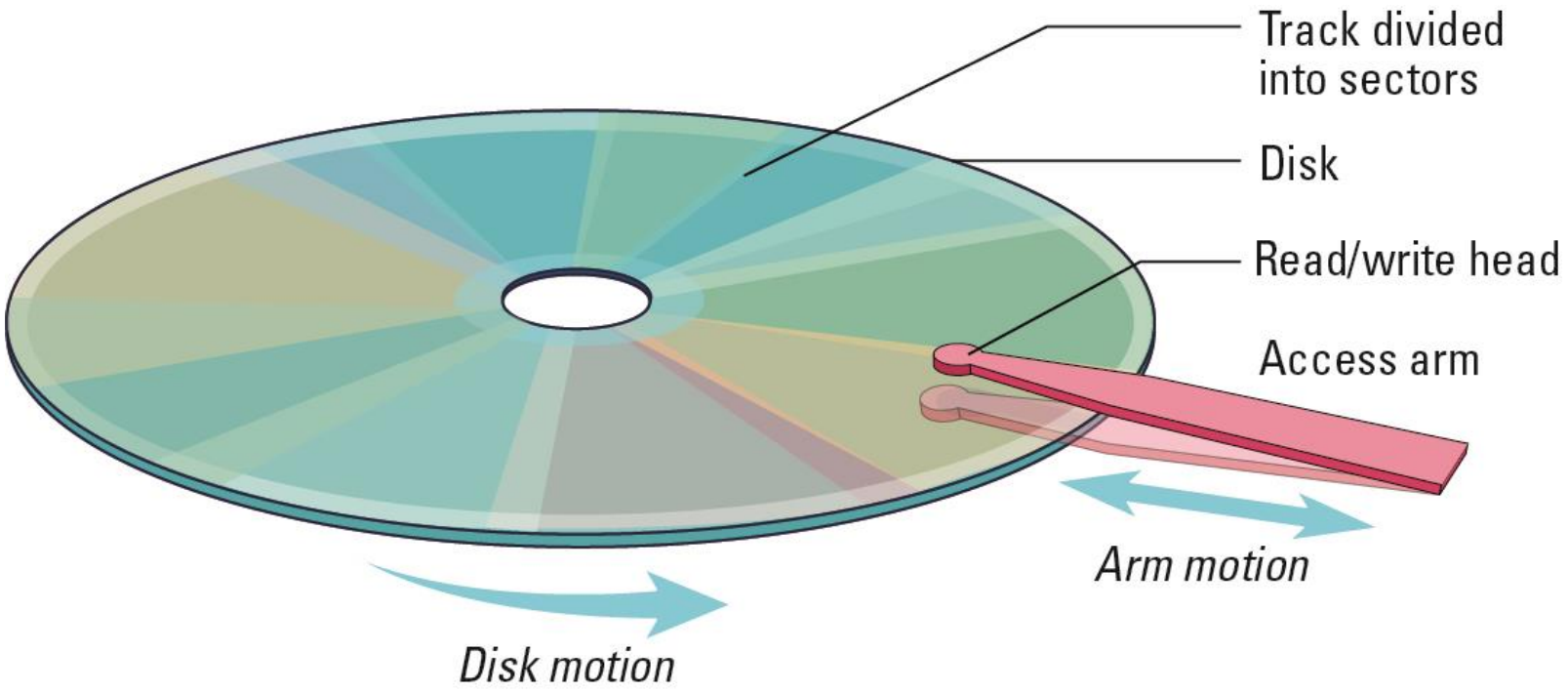
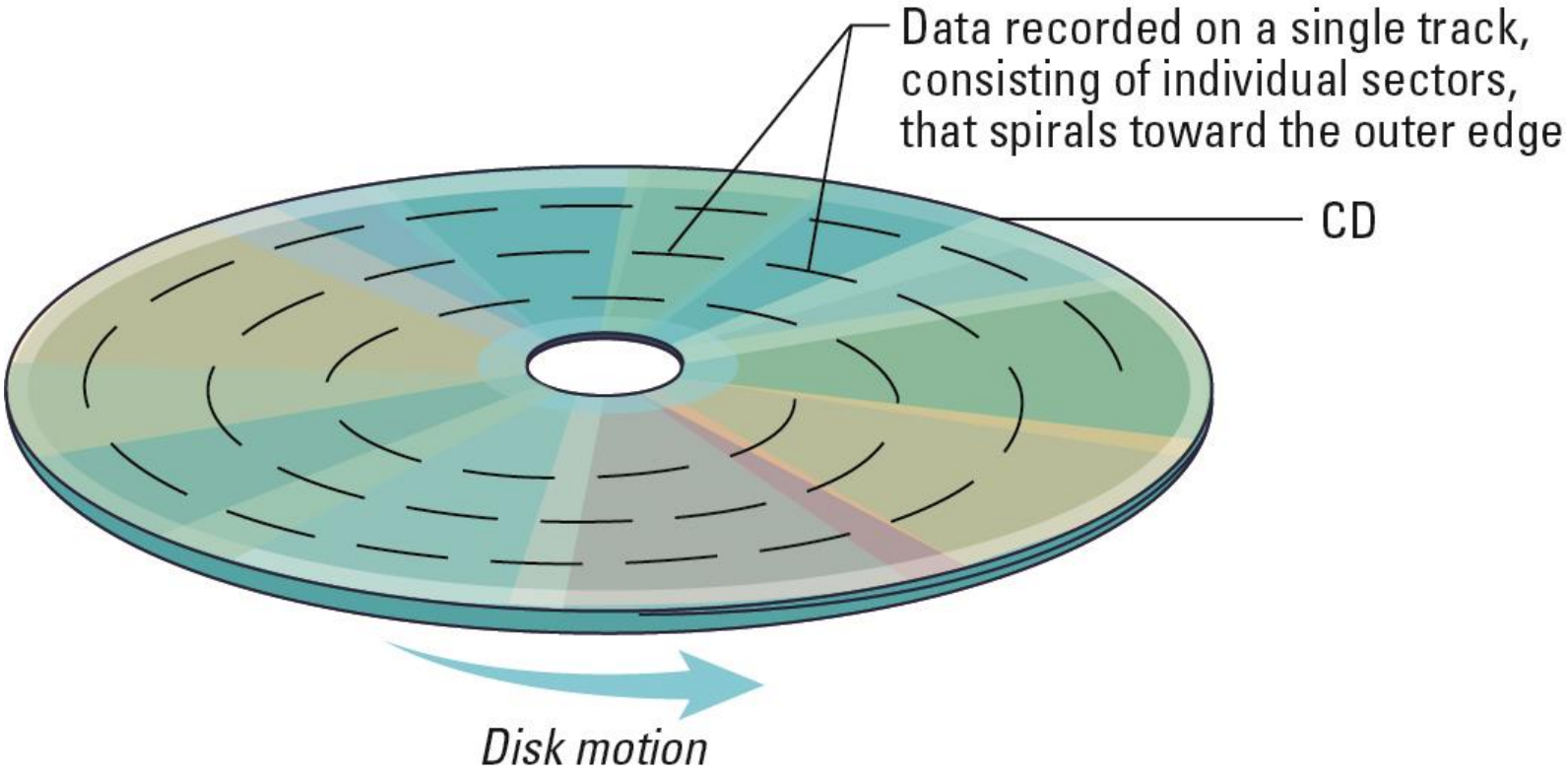


Figure 1.10 CD storage



Flash Drives

- **Flash Memory** – circuits that traps electrons in tiny silicon dioxide chambers
- Repeated erasing slowly damages the media
- Mass storage of choice for:
  - Digital cameras
  - Smartphones
- **SD Cards** provide GBs of storage

1.4 Representing Information as Bit Patterns

- Many different kinds of information can be encoded as bit patterns
- Systems for encoding information have been established for
  - Text
  - Numeric Data
  - Images
  - Sound
  - Other data

Representing Text

- **Each character (letter, punctuation, etc.) is assigned a unique bit pattern.**
  - **ASCII**: Uses patterns of 7-bits to represent most symbols used in written English text
  - **ISO** developed a number of 8 bit extensions to ASCII, each designed to accommodate a major language group
  - **Unicode**: Uses patterns up to 21-bits to represent the symbols used in languages world wide, 16-bits for world's commonly used languages

Representing Text

USASCII code chart

Bits b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4 b_3 b_2 b_1 ↓ Row / Column →	0	1	2	3	4	5	6	7
0 0 0 0	0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2	STX	DC2	"	2	B	R	b	r
0 0 1 1	3	ETX	DC3	#	3	C	S	c	s
0 1 0 0	4	EOT	DC4	$	4	D	T	d	t
0 1 0 1	5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6	ACK	SYN	&	6	F	V	f	v
0 1 1 1	7	BEL	ETB	'	7	G	W	g	w
1 0 0 0	8	BS	CAN	(	8	H	X	h	x
1 0 0 1	9	HT	EM	)	9	I	Y	i	y
1 0 1 0	10	LF	SUB	*	:	J	Z	j	z
1 0 1 1	11	VT	ESC	+	;	K	[	k	{
1 1 0 0	12	FF	FS	,	<	L	\	l	|
1 1 0 1	13	CR	GS	-	=	M	]	m	}
1 1 1 0	14	SO	RS	.	>	N	^	n	~
1 1 1 1	15	SI	US	/	?	O	_	o	DEL

Representing Text

ISO/IEC 8859十五個字符集的比較

Bin	Oct	Dec	Hex	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16
10100000	240	160	A0	*NBSP*														
10100001	241	161	A1	¡	Ą	Ħ	Ą	Ë		˝		¡	Ą	ก	˛	Ḃ	¡	Ą
10100010	242	162	A2	¢	˘	˘	ĸ	Ъ		˛	¢	¢	Ē	ข	¢	ḃ	¢	ą
10100011	243	163	A3	£	Ł	£	Ŗ	Ґ		£	£	£	Ģ	ฃ	£	£	£	Ł
10100100	244	164	A4	¤	¤	¤	¤	Є	¤	€	¤	¤	Ī	ค	¤	Ċ	€	€
10100101	245	165	A5	¥	Ľ		Ĩ	S		Ð	¥	¥	Ĩ	ฅ	„	ċ	¥	„
10100110	246	166	A6	¦	Ś	Ĥ	Ļ	І		¦	¦	¦	Ķ	ฆ	¦	Ḋ	Š	Š
10100111	247	167	A7	§	§	§	§	Ї		§	§	§	§	ง	§	§	§	§
10101000	250	168	A8	¨	¨	¨	¨	J		¨	¨	¨	Ļ	จ	Ø	Ẁ	š	š
10101001	251	169	A9	©	Š	İ	Š	Љ		©	©	©	Đ	ฉ	©	©	©	©
10101010	252	170	AA	ª	Ş	Ş	Ē	Њ		ͺ	×	ª	Š	ช	Ŗ	Ẃ	ª	Ș
10101011	253	171	AB	«	Ť	Ğ	Ģ	Ћ		«	«	«	Ŧ	ซ	«	ḋ	«	«
10101100	254	172	AC	¬	Ź	Ĵ	Ŧ	Ќ	،	¬	¬	¬	Ž	ฌ	¬	Ỳ	¬	Ź
10101101	255	173	AD											ญ				
10101110	256	174	AE	®	Ž		Ž	Ў			®	®	Ū	ฎ	®	®	®	ź
10101111	257	175	AF	¯	Ż	Ż	¯	Ų		—	¯	¯	Ŋ	ฏ	Æ	Ÿ	¯	Ż
10110000	260	176	B0	°	°	°	°	А		°	°	°	°	ฐ	°	Ḟ	°	°
10110001	261	177	B1	±	ą	ħ	ą	Б		±	±	±	ą	ฑ	±	ḟ	±	±
10110010	262	178	B2	²	˛	²	˛	В		²	²	²	ē	ฒ	²	Ġ	²	Č
10110011	263	179	B3	³	ł	³	ŗ	Г		³	³	³	ģ	ณ	³	ġ	³	ł

Representing Text

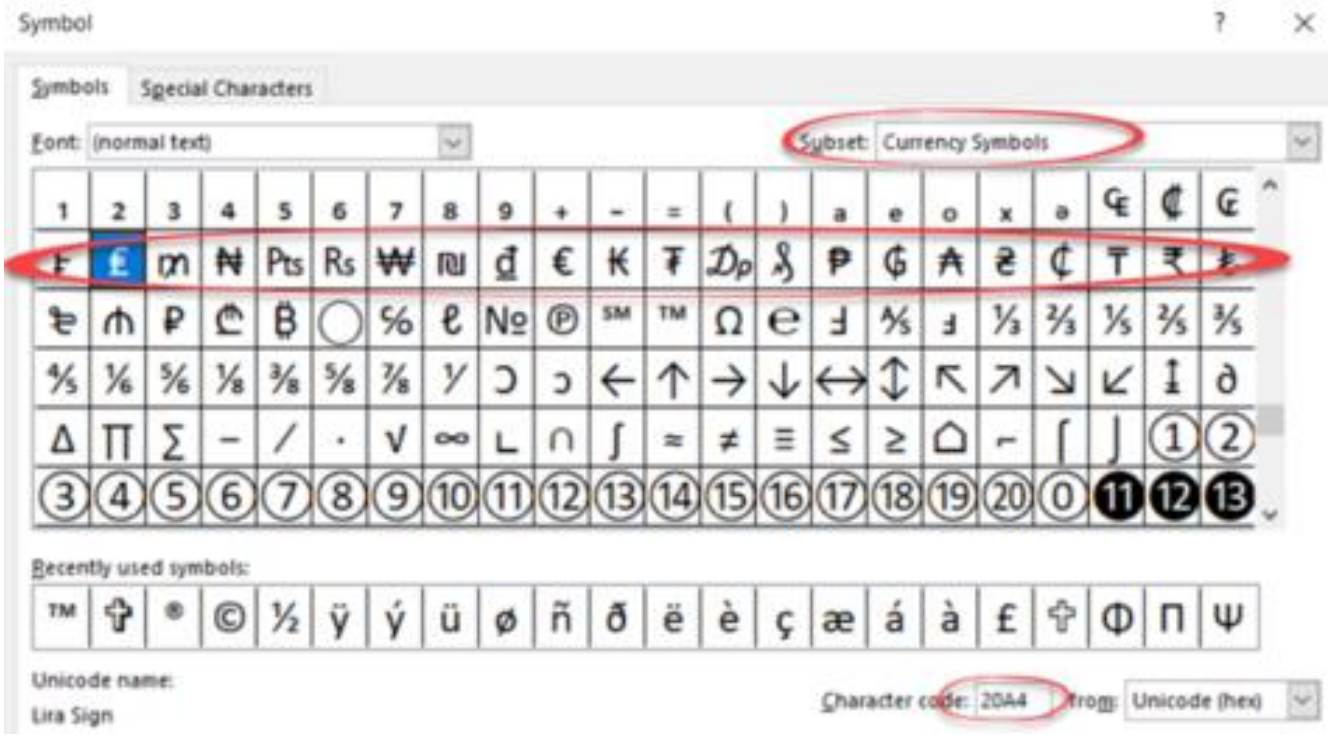
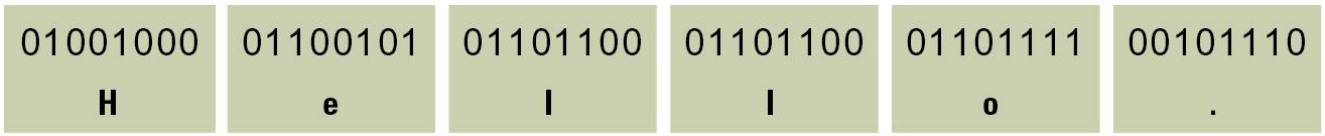


Figure 1.11 The message “Hello.” in ASCII or UTF-8 encoding



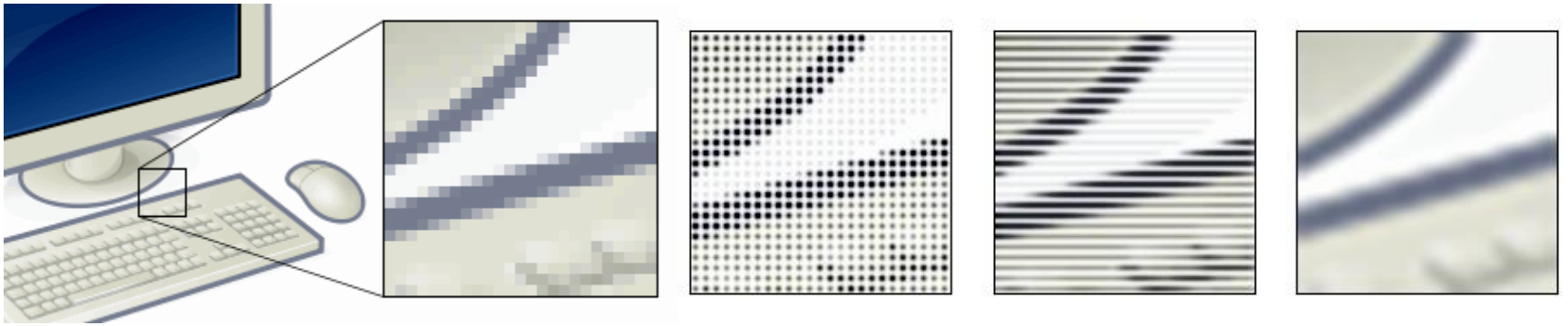
Representing Numeric Values

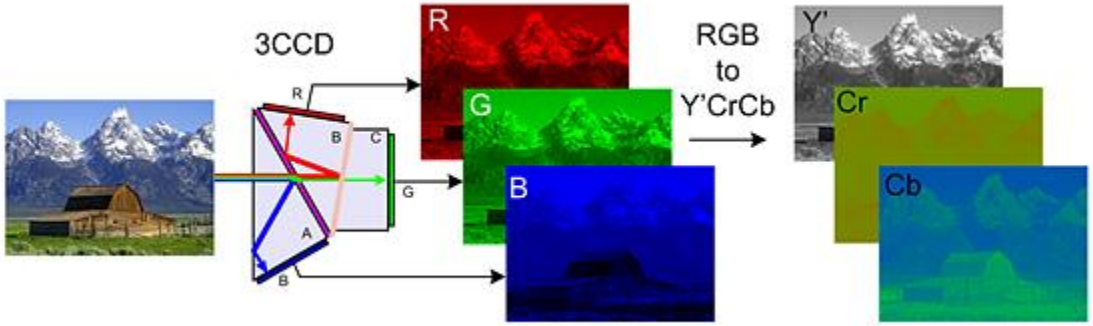
- **Binary notation**: Uses bits to represent a number in base two
  - All numeric values in a computer are stored in sequences of 0s and 1s
  - Counting from 0 to 8:
    - 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000

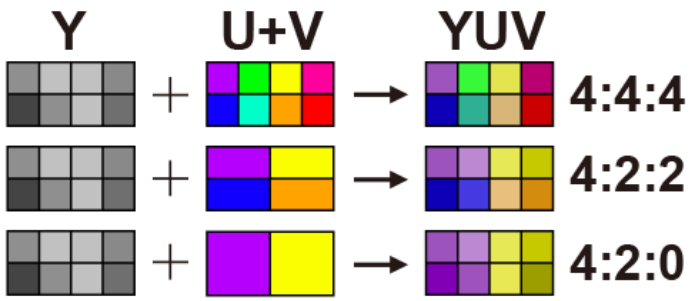
Representing Images

- Bit map techniques
  - Pixel: “picture element” represents one dot
  - RGB: Red, Green, and Blue components
  - Luminance and chrominance
  - Problems with scaling up images
- Vector techniques
  - Represent images with geometric structures
  - Scalable
  - TrueType and PostScript

Representing Images



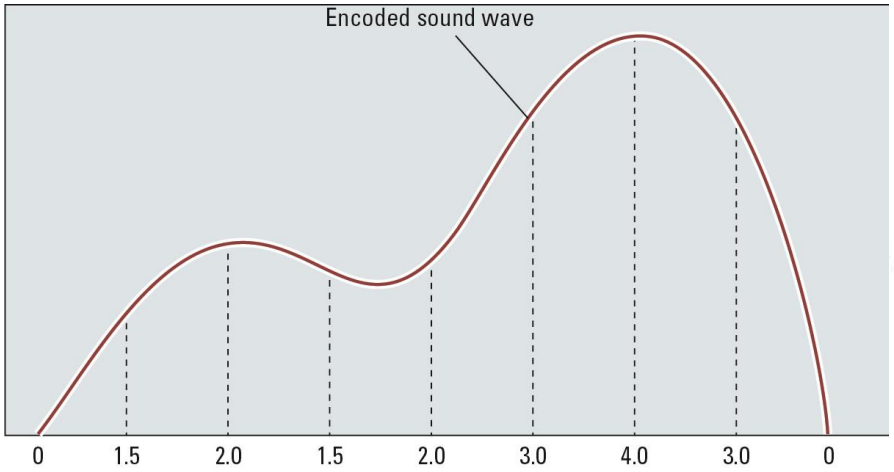


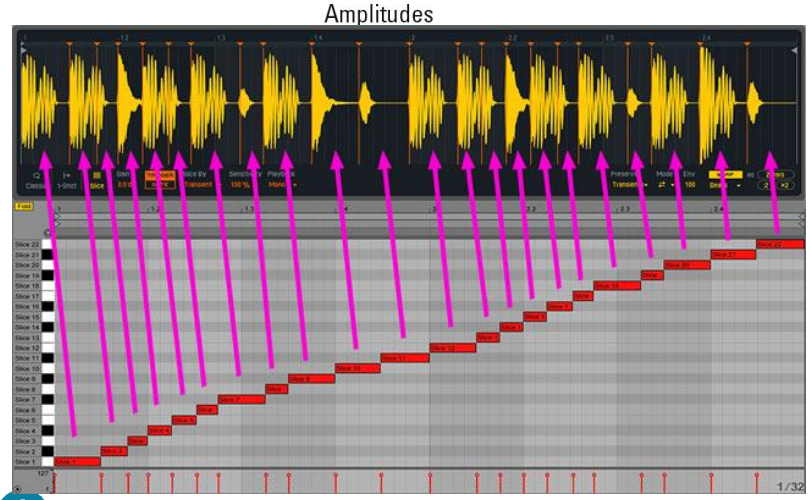


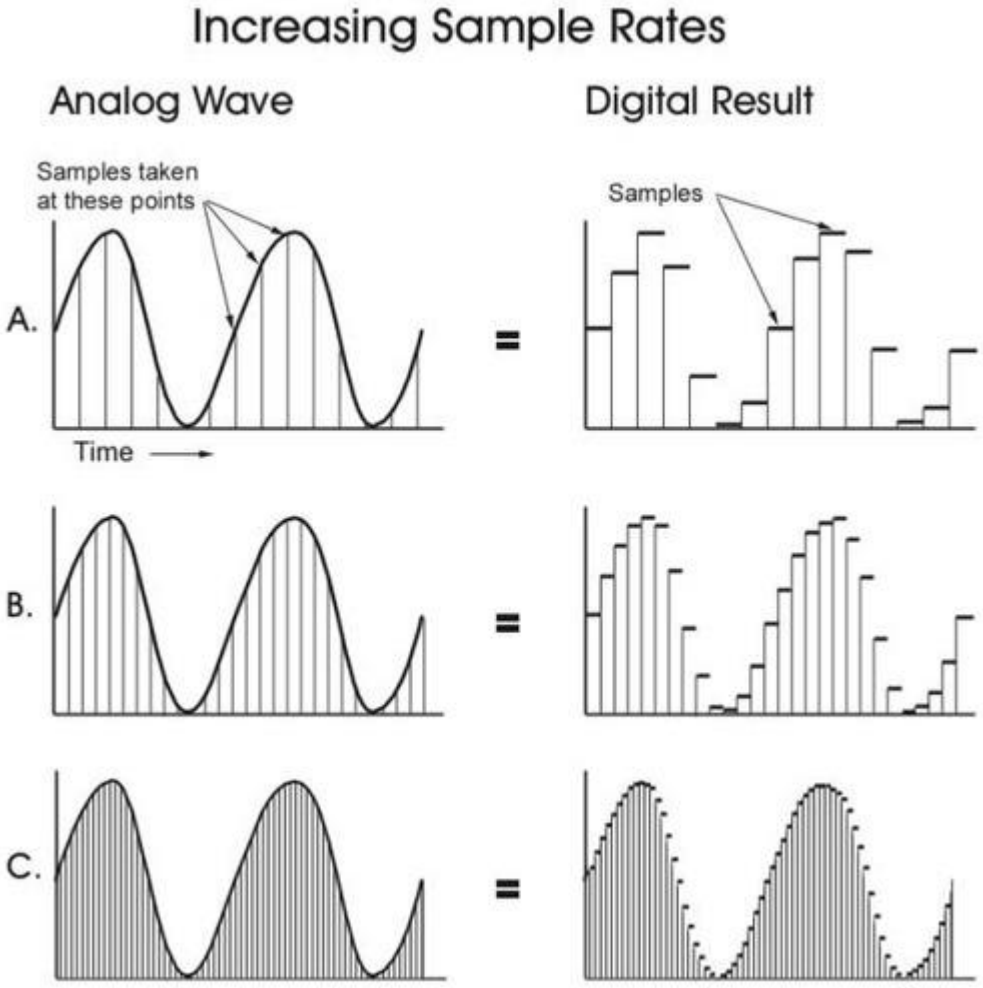
Representing Sound

- Sampling techniques that record actual audio
  - Long-distance telephone: 8000 samples/sec
  - CD sound: 44,100 samples/sec
- MIDI stores directions for making sound
  - Used in music synthesizers
  - Encodes which instrument, note, and duration

Figure 1.12 The sound wave represented by the sequence 0, 1.5, 2.0, 1.5, 2.0, 3.0, 4.0, 3.0, 0







1.5 The Binary System

The traditional decimal system is based on powers of ten.

The Binary system is based on powers of two.

Figure 1.13 The base ten and binary systems

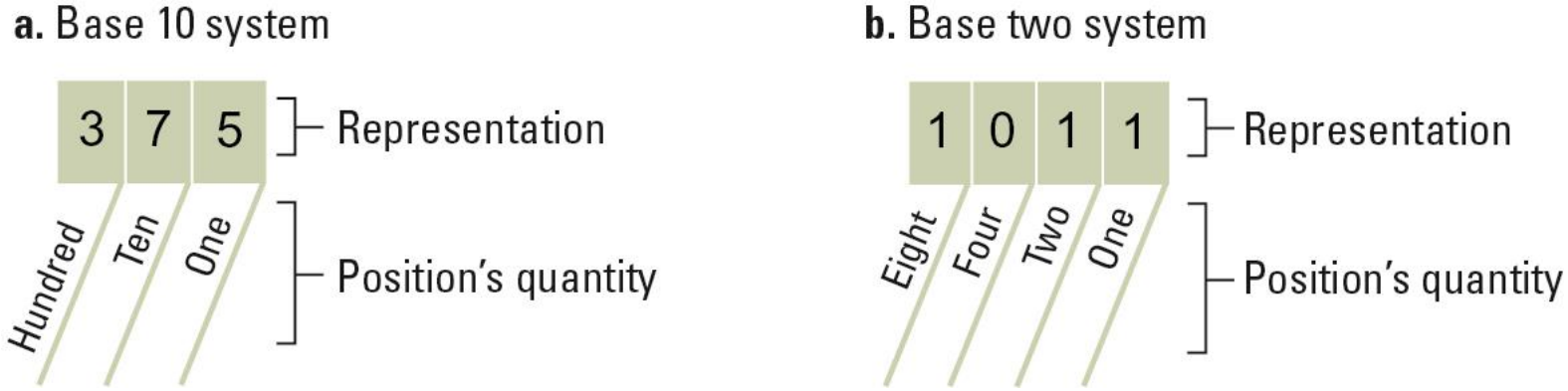


Figure 1.14 Decoding the binary representation 100101

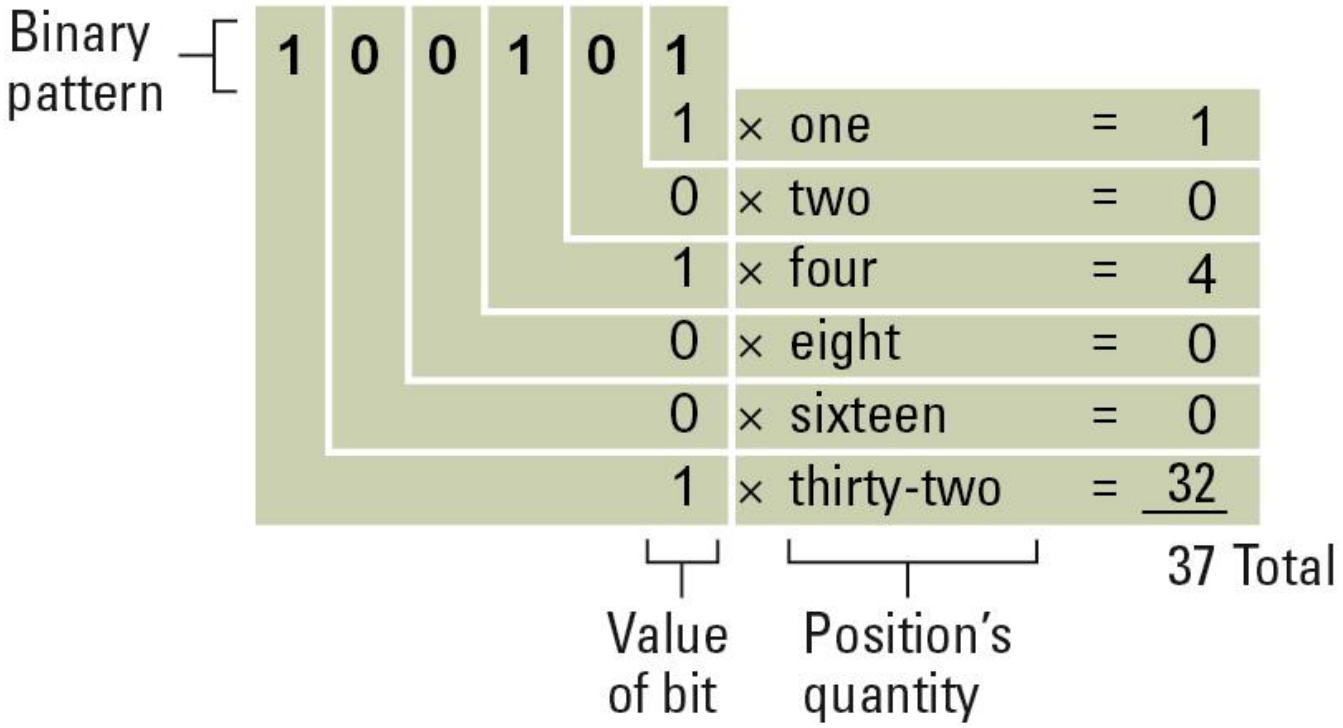


Figure 1.15 An algorithm for finding the binary representation of a positive integer

Step 1. Divide the value by two and record the remainder.

Step 2. As long as the quotient obtained is not zero, continue to divide the newest quotient by two and record the remainder.

Step 3. Now that a quotient of zero has been obtained, the binary representation of the original value consists of the remainders listed from right to left in the order they were recorded.

Figure 1.16 Applying the algorithm in Figure 1.15 to obtain the binary representation of thirteen

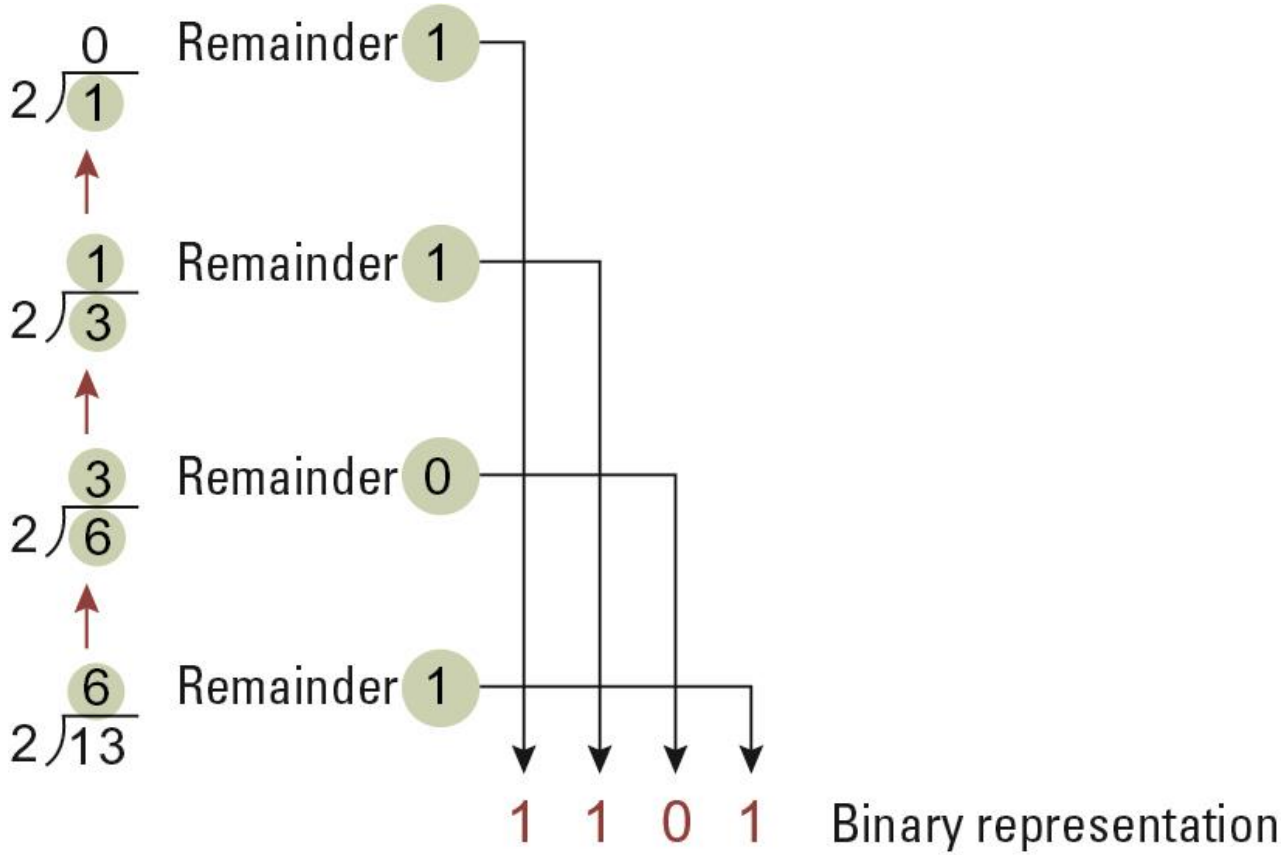


Figure 1.17 The binary addition facts

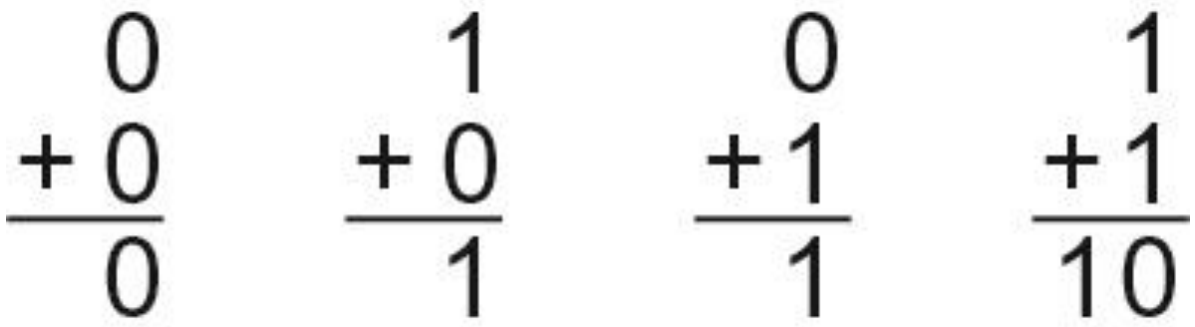
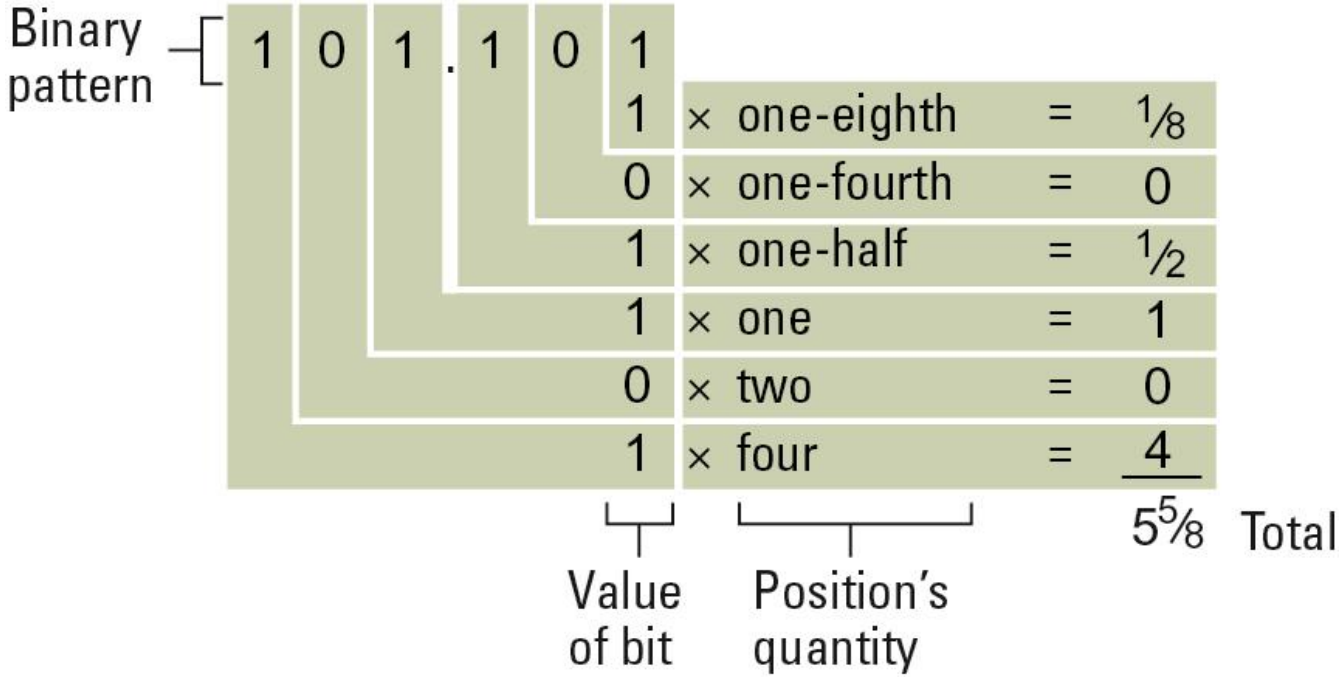


Figure 1.18 Decoding the binary representation 101.101



1.6 Storing Integers

- **Two's complement notation:** The most popular means of representing integer values
- **Excess notation:** Another means of representing integer values

Figure 1.19 Two's complement notation systems

a. Using patterns of length three

Bit pattern	Value represented
011	3
010	2
001	1
000	0
111	–1
110	–2
101	–3
100	–4

b. Using patterns of length four

Bit pattern	Value represented
0111	7
0110	6
0101	5
0100	4
0011	3
0010	2
0001	1
0000	0
1111	–1
1110	–2
1101	–3
1100	–4
1011	–5
1010	–6
1001	–7
1000	–8

Figure 1.20 Coding the value -6 in two's complement notation using four bits

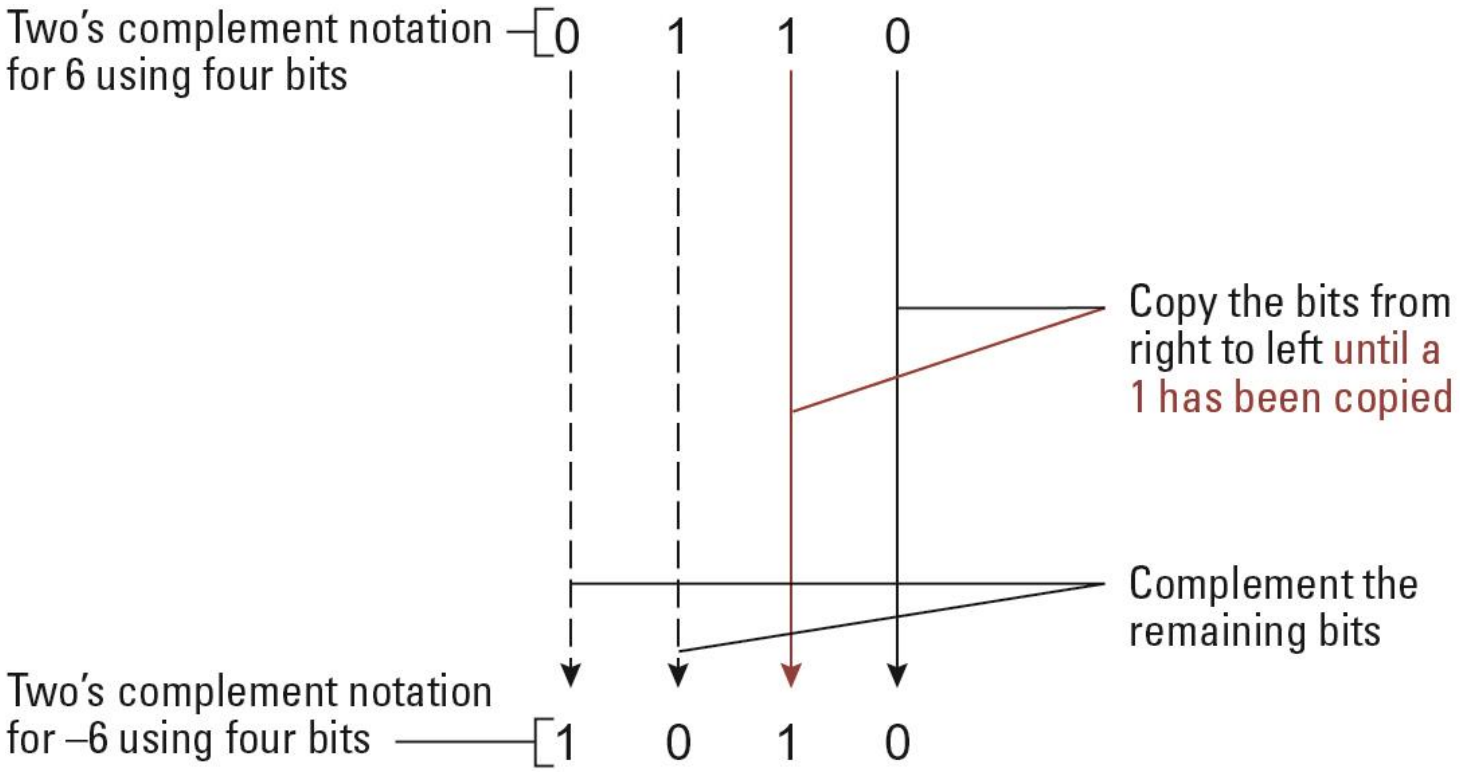
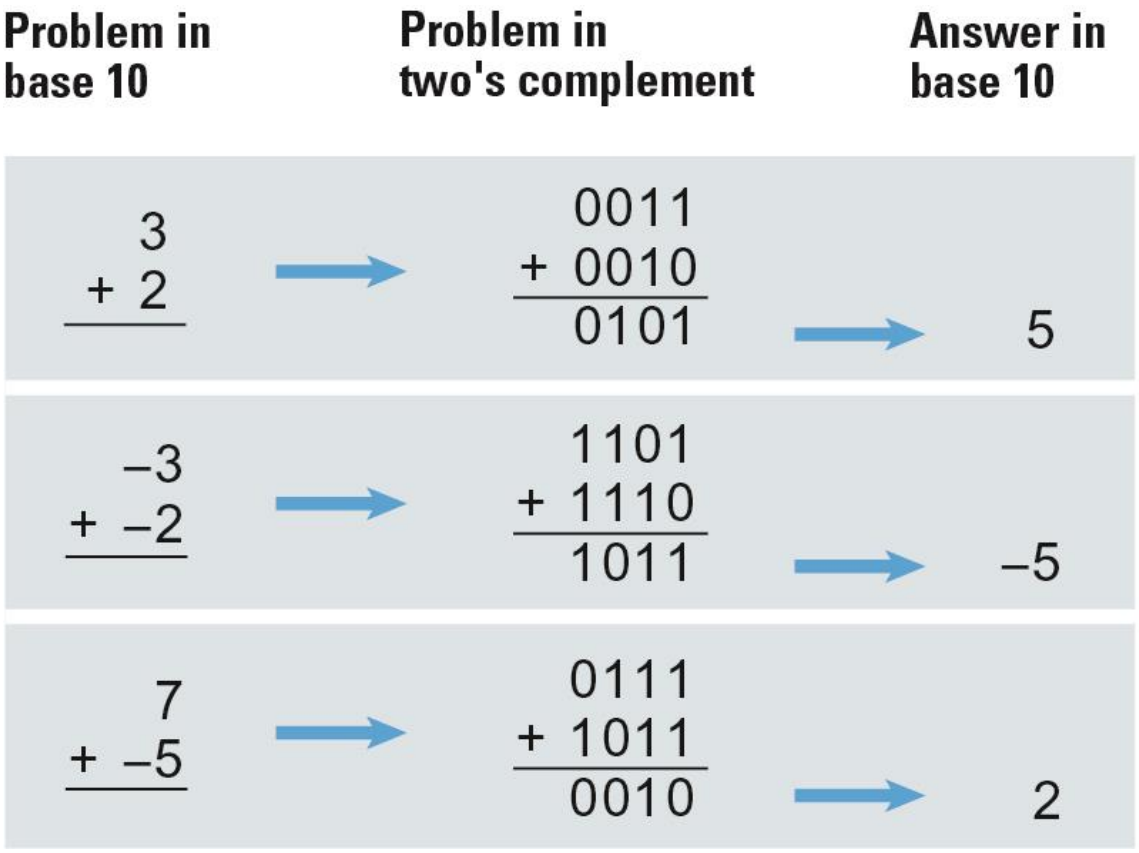


Figure 1.21 Addition problems converted to two's complement notation



The Problem of Overflow

- There is a limit to the size of the values that can be represented in any system
- Overflow
  - occurs when a computation produces a value that falls outside the range of values that can be represented in the machine
  - If the resulting sign bit is incorrect, an overflow has occurred
  - 16 bit systems have been upgraded to 32 bit systems

Figure 1.22 An excess eight conversion table

Bit pattern	Value represented
1111	7
1110	6
1101	5
1100	4
1011	3
1010	2
1001	1
1000	0
0111	−1
0110	−2
0101	−3
0100	−4
0011	−5
0010	−6
0001	−7
0000	−8

Figure 1.23 An excess notation system using bit patterns of length three

Bit pattern	Value represented
111	3
110	2
101	1
100	0
011	–1
010	–2
001	–3
000	–4

1.7 Storing Fractions

- **Floating-point Notation:** Consists of a sign bit, a mantissa field, and an exponent field.
  - Normalized form: fill the mantissa starting with the left-most 1

Figure 1.24 Floating-point notation components

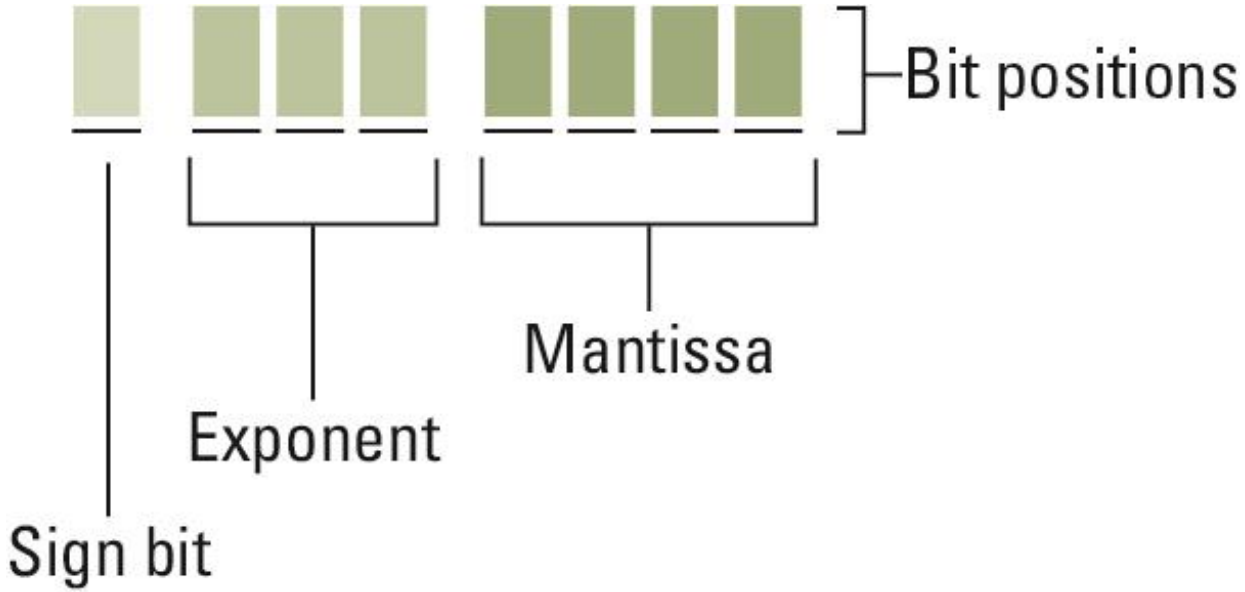
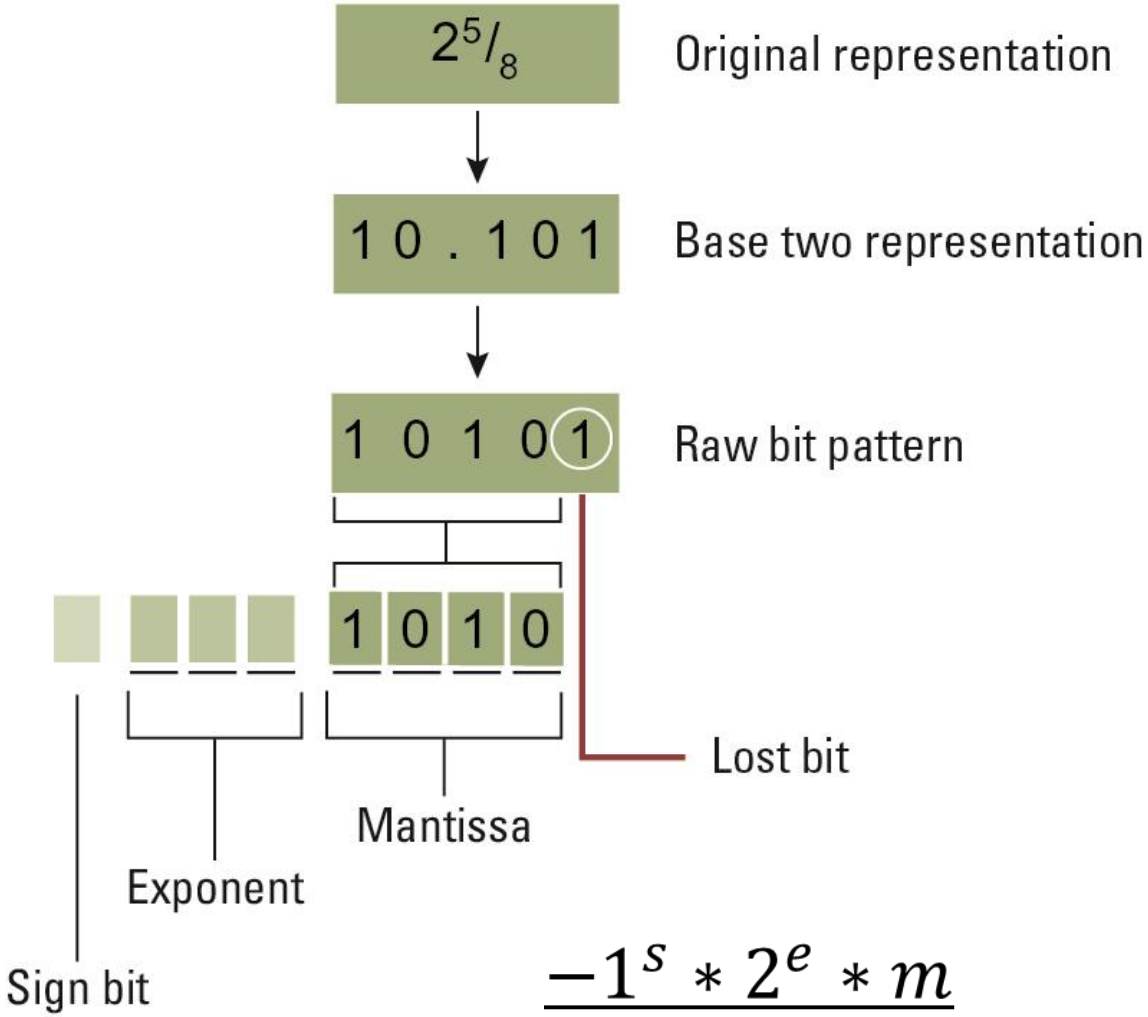


Figure 1.25 Encoding the value 2 5/8



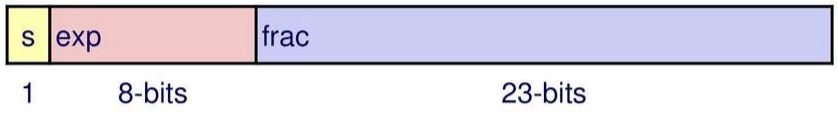
- IEEE Standard 754
  - Most (but not all) computer manufactures use IEEE-754 format
  - Number represented:

$$(-1)^S * (1.M)*2^{(E - Bias)}$$

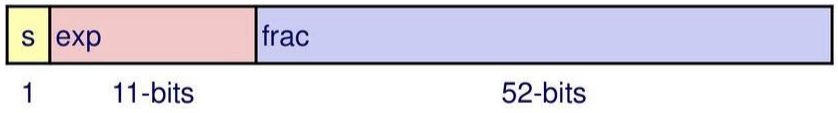
2 main formats: single and double

IEEE Standard 754: Precision options

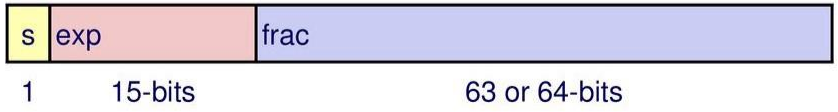
Single precision: 32 bits



Double precision: 64 bits



Extended precision: 80 bits (Intel only)



Truncation (Round-off) Errors

- Occur when part of the value being stored is lost because the mantissa is not large enough
- Non-terminating expansions of fractions
  - This happens more often with binary notation
  - The value of one-tenth cannot be stored exactly binary notation
  - Often these values are converted to integers

Numerical Analysis

- The study of dealing with problems when computing large values that require significant accuracy
- The order in which values are added can lead to two different results
- Adding very small values to very large values can result in errors

1.8 Data and Programing

A ***programming language*** is a computer system created to allow humans to precisely express algorithms using a higher level of abstraction.

Getting Started with Python

- ***Python:*** a popular programming language for applications, scientific computation, and as an introductory language for students
- Freely available from **www.python.org**
- Python is an *interpreted language*
  - Typing:

```
print('Hello, World!')
```

  - Results in:

```
Hello, World!
```

Variables

- **Variables:** name values for later use
- Analogous to mathematic variables in algebra

```
s = 'Hello, World!'
print(s)

my_integer = 5
my_floating_point = 26.2
my_Boolean = True
my_string = 'characters'
my_integer = 0xFF
```

Operators and Expressions

```
print(3 + 4)     # Prints 7
print(5 – 6)     # Prints -1
print(7 * 8)     # Prints 56
print(45 / 4)    # Prints 11.25
print(2 ** 10)   # Prints 1024

s = 'hello' + 'world'
s = s * 4
print(s)
```

Currency Conversion

```
# A converter for currency exchange.

USD_to_GBP = 0.66    # Today's exchange rate
GBP_sign = '\u00A3' # Unicode value for £
dollars = 1000       # Number dollars to convert

# Conversion calculations
pounds = dollars * USD_to_GBP

# Printing the results
print('Today, $' + str(dollars))
print('converts to ' + GBP_sign + str(pounds))
```

Debugging

- *Syntax errors*

```
print(5 +)
SyntaxError: invalid syntax

pront(5)
NameError: name 'pront' is not defined
```

- *Semantic errors*
  - Incorrect expressions like

```
total_pay = 40 + extra_hours * pay_rate
```

- *Runtime errors*
  - Unintentional divide by zero

1.9 Data Compression

- Lossy versus lossless
- Run-length encoding
- Frequency-dependent encoding

  (Huffman codes)
- Relative encoding
- Dictionary encoding (Includes adaptive dictionary encoding such as LZW encoding.)

Compressing Images

- GIF (Graphics Interchange Format): Good for cartoons
- JPEG (Joint Photographic Experts Group): Good for photographs
- TIFF (Tag Image File Format): Good for image archiving

Compressing Audio and Video

- MPEG (Moving Picture Experts Group)
  - High definition television broadcast
  - Video conferencing
- MP3 (ISO-MPEG Audio Layer-3)
  - Temporal masking
  - Frequency masking

1.10 Communication Errors

- Goal: To reduce errors and increase the reliability of computing equipment
- Parity bits (even versus odd)
- Checkbytes
- Error correcting codes
  - Hamming Distance

Figure 1.26 The ASCII codes for the letters A and F adjusted for odd parity

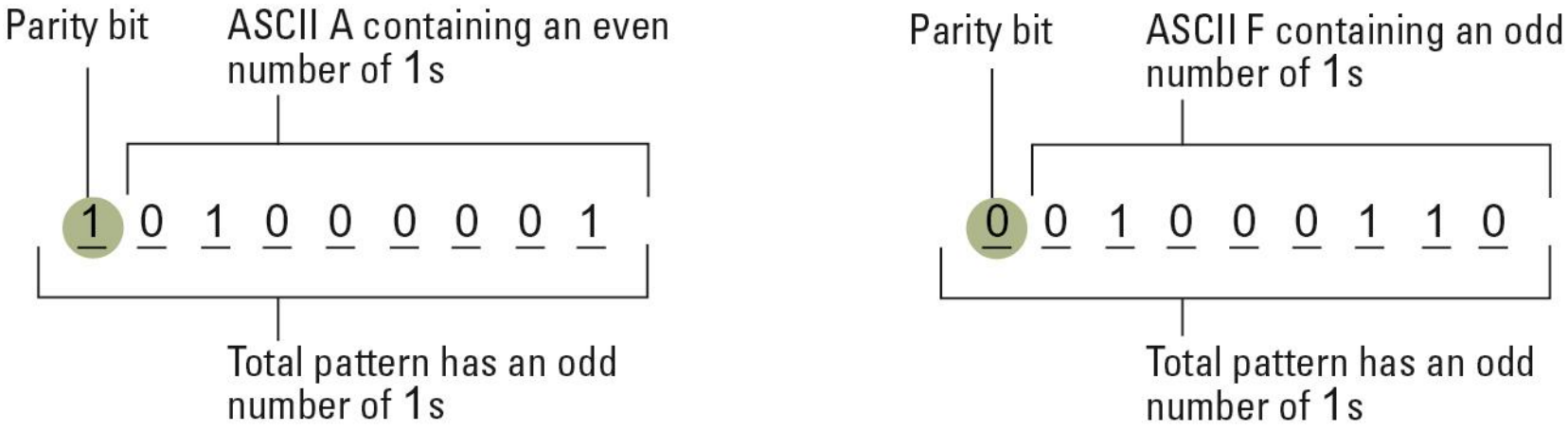


Figure 1.27 An error-correcting code

Symbol	Code
A	000000
B	001111
C	010011
D	011100
E	100110
F	101001
G	110101
H	111010

Figure 1.28 Decoding the pattern 010100 using the code in Figure 1.27

Character	Code	Pattern received	Distance between received pattern and code
A	0 0 0 0 0 0	0 **1** 0 **1** 0 0	2
B	0 0 1 1 1 1	0 **1** **0** 1 **0** **0**	4
C	0 1 0 0 1 1	0 1 0 **1** **0** **0**	3
D	0 1 1 1 0 0	0 1 **0** 1 0 0	**1** — Smallest distance
E	1 0 0 1 1 0	**0** **1** 0 1 **0** 0	3
F	1 0 1 0 0 1	**0** **1** **0** **1** 0 **0**	5
G	1 1 0 1 0 1	**0** 1 0 1 0 **0**	2
H	1 1 1 0 1 0	**0** 1 **0** **1** **0** 0	4